Raytheon
BBN Technologies

ONR STEM Grand Challenge
Extensible Adaptive System for STEM Learning

Contract # N00014-12-C-0535
Raytheon BBN Technologies Corp. (BBN) Reference # 14217
In fulfillment of contract deliverable items # A005

System Design / Architecture Report

Revised on April 12, 2013

BBN Technical POC:	**Distribution:**
Premkumar Natarajan (PI) Raytheon BBN Technologies 10 Moulton St. Cambridge, MA 02138-1119 Ph: (617) 873-5472 E-Mail: prem@bbn.com	Office of Naval Research 875 North Randolph St. Attn: Joseph Cohn ONR Code:341 Arlington, VA 22203-1995 Ref: Contract N00014-12-C-0535 E-Mail: joseph.cohn@navy.mil **Cc:** ACO Timothy Keenan DCMA, BOSTON (Boston) Ph: (617) 753-3895 Email: Timothy.Keenan@dcma.mil

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
APR 2013		**00-00-2013 to 00-00-2013**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
ONR STEM Grand Challenge:Extensible Adaptive System for STEM Learning	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Raytheon BBN Technologies,10 Moulton St.,Cambridge,MA,02138**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**11**	

Table of Contents

1 Introduction

This is the System Design / Architecture Report (CDRL Data Item #A005) for EAITS, an extensible and adaptive STEM Learning System that BBN is developing under the ONR STEM Grand Challenge (GC) Program, Contract # N00014-12-C-0535. EAITS embodies algorithmic research and system development for high-school and college level advanced learning systems. These learning systems employ state-of-the-art pedagogical and computation techniques to improve student proficiency in targeted subject areas. For the ONR STEM GC Program, BBN will develop a prototype version of EAITS and demonstrate its customization to the SAT Physics curriculum. In this document, we first present a conceptual overview of the EAITS design. The second part of this document describes the architectural blueprint that is used to organize the various system components that constitute the EAITS.

2 Conceptual Overview

Note: This overview is shared with the System Requirements and Concept Report (SRCR).

The EAITS will advance the state-of-the-art in learning technologies through the fusion of multiple forms of adaptive support with highly effective learning activities. Each of these forms of support and activities has been individually shown to be effective in terms of improving learning outcomes in prior work. Through integration of these complementary forms of support and careful coordination achieved by statistical models, we can close the gap between the efficacy of automated learning systems that are scalable for mass use and the 2σ efficacy of scarcely available human tutors.

The SAT Physics version of EAITS will be an advanced learning system that allows high-school students to improve their proficiency at solving SAT-style physics problems. The system achieves this through two learning activities. First, EAITS will allow students to engage in learning through problem solving. Second, students can explore multimedia instructional content such as videos about Physics concepts. Students can also participate in both of these learning activities in a self-paced, anytime, anywhere manner, using a variety of devices and platforms that have a standard web-browser (e.g., Internet Explorer, Google Chrome, or Mozilla Firefox).

The learning efficacy of this system will be amplified through the use of personalized adaptive tutoring that is integrated in both the learning activities described above. The tutor will use comprehensive student modeling in combination with a domain model and rigorous analysis of the student's actions in the learning environment to constantly update and adapt its beliefs about the student's proficiencies and preferences. For example, if a student is unable to solve a problem involving the concept of *resistivity*, the tutor updates it belief about the student's proficiency in this concept and may decide to initiate interventions that will lead to improvement in the student's knowledge. The EAITS tutor will be capable of using multiple forms of support to tailor the interventions delivered to the student including the use of corrective, reflective, and engagement support. Furthermore, the intervention presented to the student will be informed by prior observations (captured in the student model). For example, a natural language dialog may be more effective for certain learners than the use of instructional videos.

The tutor also will use a data-driven assessment model to estimate and predict the expected performance of a student on standardized tests within the topics covered by the system. This assessment capability will serve as a regulatory mechanism to balance between tutor and student initiative during learning activities.

In addition to the SAT Physics version of EAITS, the underlying platform will provide authoring tools that support development and maintenance of high-quality learning content, which can be used within EAITS. These tools can be used by domain experts and system administrators to add and modify knowledge sources used by the SAT Physics EAITS in an ongoing manner without the need to modify the underlying software platform.

3 System Architecture

This report is a working document. In addition to serving as a reference blueprint for the development of EAITS during the course of the project, it captures ongoing discussions between ONR and BBN on the EAITS software architecture.

Figure 1 shows the organization of EAITS components. These components are broadly divided into three categories: End User Interfaces, Web Services, and Resources. This division is based on several aspects including technology, consumer, interaction with other components, and maintenance. The rest of this section describes the components under each of these categories.

3.1 End User Interface

End users include learners, content developers, subject matter experts, and system administrators. All such users access the system through one of two end user interfaces, which are both accessible over the web using standard browsers. We are using cross-platform technologies such as Javascript and CSS to develop these interfaces. There are two distinct interfaces for two broad categories of user roles, the learner and the subject matter expert/administrator (jointly referred to as the author).

3.1.1 Learning Environment

End users with a student/learner role are directed to the learning environment after login. The learning environment is an extensible graphical user interface (GUI) that runs concurrent applications necessary for supporting the interactive learning activities. A screenshot of a prototype implementation of the learning environment is shown in Figure 2. Four applications reside within this learning environment.

1. **Communication:** This text-based instant messaging application allows natural language interaction between the learner and the system (playing the role of a tutor). It is used for tutorial dialog, learner initiated help requests and system-initiated engagement prompts. This application is docked in the learning environment. It is not closeable but the user is able to minimize/hide this application.
2. **Student Profile:** This is another docked application that provides a summary of the user profile including a display of user information, assessments, and notifications. User and Session operations such as *settings* & *logout* are supported through this application.

3. **Content Exploration:** The content exploration application allows users to navigate through the course curriculum and instructional materials (including problems, dialogs, videos, etc.). This application is part of the main tab panel in the learning environment and is not closeable.
4. **Problem Solving:** This application allows learners to interact with the primary learning activity of our system. A new application tab (in the main tab panel) is launched for every problem a student is solving. This application is closeable.

A configuration object associated with each user maintains the state of the learning environment between learning session. Furthermore, the learning environment maintains a long-poll based HTTP data-link [1] with the middleware services which is used to receive server pushed event notifications which are then distributed to the relevant applications for processing.

The learning environment is designed to be used on desktop as well as touch-screen tablet devices. To accommodate a touch-screen keyboard at the bottom of the screen, important areas of user attention (e.g. input fields) are placed towards the top of the screen.

3.1.2 Workbench

Authors are end users who access the system through the workbench. The workbench is an extensible GUI that is comprised of tools for content authoring and system administration.

1. **Administration/Configuration Tool** is used to administer users, data, and service components. It provides basic utilities for managing the system. It is specifically designed for use during laboratory experiments (such as EAITS evaluations) to monitor students. Furthermore, some system configurations, such as the availability of certain types of support, are provided through this tool.
2. **Domain Model Editor** allows subject matter experts (SME) to represent the learning domain as a collection of chapters, concepts, problems, dialogs and other instructional units. SMEs are allowed to add, edit, remove and link these units to each other and specify their associations.
3. **Dialog Script Authoring Tool** support authoring tutorial dialog scripts. Integrated libraries of interaction concepts (e.g., generic feedback, common responses, etc.) support efficient dialog development.
4. **Problem & Tutor Model Editor** is a combination of two tools. First, a WYSIWYG UI editor that is used to create learning problems and its sub-step scaffolding that represents the decomposition of the problem into simple steps. Second, an advanced model editor allows content developers to create model tracing tutoring support for sub-steps. Model tracing tutors [2] track student actions during problem solving and provide appropriate help and feedback to the student.

The key design principle for each of the tools is support for creation of high-quality content. Reducing the cost of development of such content is a secondary goal. These tools are designed for advanced users who will be expected to have some training to use these tools. Furthermore, several quality assurance processes (automated as well as manual) and automated content processing capabilities will be integrated within these tools to harmonize the two competing principles.

Similar to the learning environment, the workbench uses a configuration object to maintain state across user-logins and an HTTP data-link to communicate with the middleware service. The workbench is designed to be used on a desktop/laptop device.

3.1.3 Shared Components

Shared component are comprised of code and resources that are shared between the Learning Environment and the Workbench. These include Javascript libraries for accessing services, Javascript UI widgets, CSS styles and media files such as icons. Some of these shared components may be hosted on third-party servers (due to licensing or ownership constraints) and are included in the user interfaces within the strictures of permitted/fair use.

3.2 Web Services

The backend web services that the UI communicates with are comprised of three component layers. First, there is a stack of platform services that provide (secure) open access to the underlying learning system/tutoring/content through well-defined application programing interfaces (API). Second, the user models are a collection of runtime objects that store data and implement event handlers associated with each user connected to the system. These two layers are discussed in the subsections below. Finally, various advanced capabilities comprise the lowest layer of stateless utilities used by event-handlers within the user models. Details of these capabilities are discussed along with their use in the user model section (§3.2.2).

3.2.1 Platform Services

A collection of secure services, accessible over HTTP, allow the UI to connect with the backend. Javascript libraries (as previously described in §3.1.3) implement well-defined APIs for accessing these services. There are three main services:

1. **Authentication services** implement best practices for validating user credentials and initializing a user model for that user. Also, the user is directed to the appropriate end-user interface (learning environment or workbench) by this service.
2. **Content and data services** implement a collection of stateless interfaces that allow authenticated users to retrieve and update various resources (described in §3.3). These include services for accessing learning content, user profiles, domain models, data logs, etc. Some of the resources may be pre-processed by the implementation of these services e.g. student profiles may be synthesized from multiple tables in the database before it is presented.
3. **Middleware services** implement long polling HTTP push interfaces that allow the UI to communicate with the interactive capabilities of the system in a two-way manner. Communication between the learning environment and the middleware use an event class as a token of information passed through this channel. Both the UI as well as the user models implement interfaces to process and produce heterogeneous events. This provides the flexibility to pass new types of events between the UI and the user model without modifying the middleware.

In addition to the services, native session management capabilities of the web servers are used to store user specific objects (such as the user model) and debug log streams.

3.2.2 User Model

The user model is comprised of a hierarchy of classes that are extended for the different user roles (student and author) supported by the system. Natively, the user model implementation provides functionality for communicating between the backend and the UI. Also, these base classes allow the management of user profiles through utility functions that access the database fields associated with users. The extensions of the base class for students and authors add a significant amount of distinct functionality, specific to each role.

For students, functionality for enabling learning applications described in §3.1.1 is provided. Specifically, communication and problem solving is supported by corresponding behavioral classes, which are tightly coupled with each other due to the frequent interaction between their behaviors. For example, dialogs launched during problem solving are made available to the learners through the communication application. A tutor object is associated with each student. Tutors are comprised of a collection of *supports*. Intelligent tutoring behaviors such as feedback during problem solving (corrective support), tutorial dialog (reflective support), social interaction strategies to regulate student attention (engagement support) and adaptive curriculum sequencing (content support) are built into the tutor object. The implementation of each intelligent capability as a form of support allows fusion of multiple forms of support by the tutor as well as tight coupling between the different forms of support to achieve harmonious coordination between their behaviors.

Advanced capabilities, which may be either learned from data or programmed by experts, are accessible by various components within the student or tutor classes (supports, communication and assessment). These capabilities are implemented in a stateless manner. For example, a text classifier used to interpret student utterances is invoked by the reflective support. The support module may attach features representing state (e.g. whether a response to a tutor question is pending) used by the text classifier. However, the classifier itself responds to every call independent of any previous or ongoing calls. This stateless design is motivated by consideration for scalability, support for bulky models (e.g. n-gram models), and the need to change models without restarting the services.

The author model class extends the user model by adding two types of objects: Locks and Tasks. Locks bind resources being modified to an author. Service side functionality for each of the tools available in the workbench (§3.1.2) is provided by one or more task objects. Tasks implement handlers for responding to events generated by each of the authoring tools. For example, a task for editing a new problem may generate request events for retrieving a problem definition, modifying the statement or image fields, adding or removing options, etc. These events are handled by the same task object. Certain tasks may invoke advanced content processing capabilities. For example, preliminary model tracing tutors may be automatically induced from logs of students' work. Capabilities such as automated quality checks, content curation, and template-based generation are implemented through this mechanism.

3.3 Resources

Resources are comprised of well-defined data/content storage technologies that form the infrastructure for the various forms of knowledge incorporated in the system. There are three resource components: Databases, a content repository, and models.

3.3.1 Databases

Databases store most of the system data with the exception of learning content and large models used by various intelligent components. The system data include user profiles, system event streams, sessions, logs and other data used for reporting/analysis of the learning process. Certain frequently used knowledge sources such as domain models (curriculum tree and concept ontology) are also stored in the database. Generally, resources only modified by the learning system (e.g., student profiles) are stored in the database.

The database schema, which is the only predetermined aspect of the database design (data is filled in during system use), is designed to be domain agnostic to facilitate the use of our learning system for new learning domains. The database is access by the services through standard database protocols. A wrapper database connection pool is employed for efficiency and scalability.

3.3.2 Content Repository

The content repository is built on top of an automated version controlled file system. It provides storage functionality for learning materials. Primarily, it is used to store content authored using our workbench, which includes problems and dialog. Associated resources such as embedded multimedia (e.g. images) and support for problems (e.g. trace graphs) are stored along with the content.

Content repository is organized to allow storage of content from multiple learning domains. Content is accessible to the services through standard file system and to the UI components through the content services discussed in §3.2.

3.3.3 Models

The models file-system stores model files used by several components in the learning system. Typically these model files (e.g. an assessment model or a triggering model for engagement support) are produced by automated machine learning algorithms trained on data extracted from the database and other sources. These models may be automatically or manually tuned by researchers before they are stored under this resource.

References

1. **HTTP Long Polling**, http://en.wikipedia.org/wiki/Push_technology#Long_polling

2. Vincent Aleven, Bruce M. McLaren, Jonathan Sewall & Kenneth R. Koedinger, 2009, **A new paradigm for intelligent tutoring systems: Example-tracing tutors**, Intl. Journal of Artificial Intelligence in Education, 19(2), 105-154

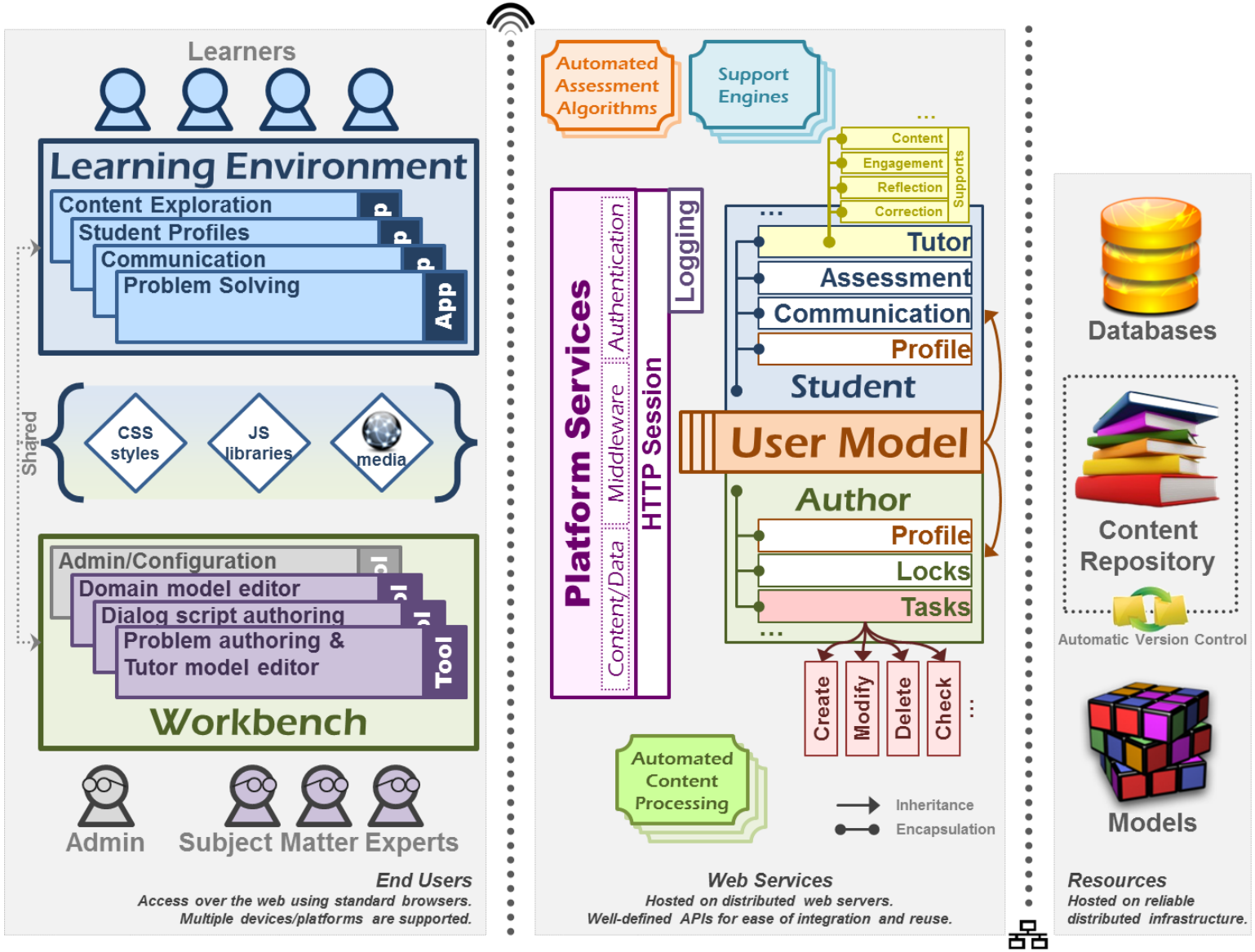

Figure 1: Block Diagram of EAITS Components

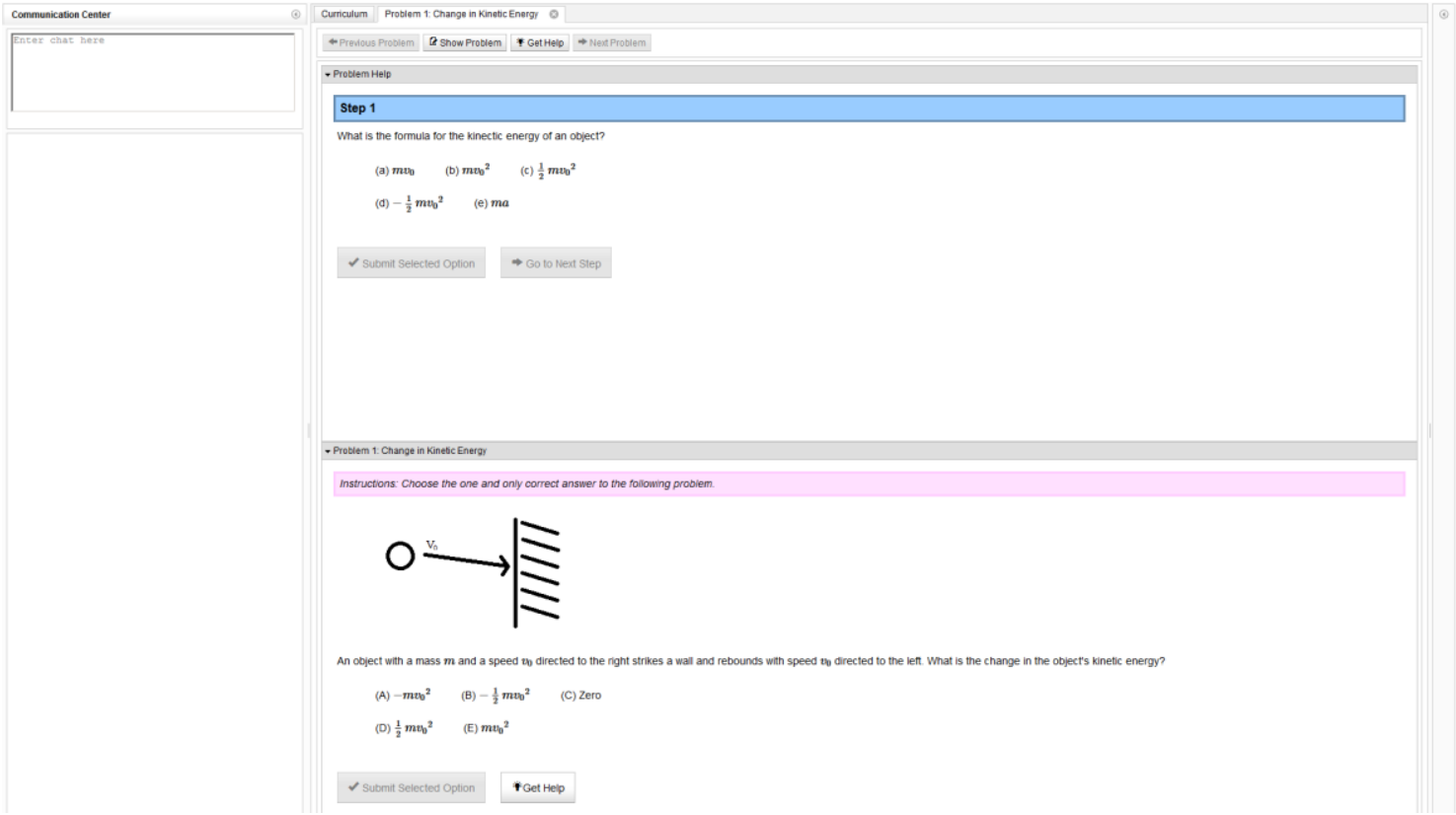

Figure 2: Screenshot of an early implementation of the Learning Environment

